Smaller Circuits for Binary Polynomial Multiplication

No Author Given

No Institute Given

Abstract. We develop a new and simple way to describe Karatsuba-like algorithms for multiplication of polynomials over $\mathbb{F}_2$. These techniques, along with interpolation-based recurrences, yield circuits that are better (smaller and with lower depth) than anything previously known. We use our optimization tools to actually build the circuits for n-term binary polynomial multiplication for values of n of practical interest.

1 Introduction

Let A, B be polynomials of degree $n-1$ over $\mathbb{F}_2$. This paper is about finding "good" circuits that compute the polynomial $A \cdot B$. We consider circuits over the basis $(\wedge, \oplus, 1)$ (that is, arithmetic over $\mathbb{F}_2$). We aim for circuits with as few gates as possible.

Applications Binary polynomial multiplication is the main operation in the arithmetic of finite fields of characteristic two. It is of particular importance in elliptic curve cryptography (see [Ber09, BGTZ08] and the references therein). Additionally finite field multiplication is used in the Galois Counter mode of operation [Dwo07]. Other important applications include binary Goppa codes and derived cryptosystems.

Our Technique We consider generalizations of the algorithm due to Karatsuba. Karatsuba's algorithm splits a polynomial into two parts and then does recursive multiplication. Researchers have already considered generalizations of this algorithm which split the input into $k \geq 3$ parts. We provide a unifying description of these generalized "Karatsuba-like" algorithms, allowing for a systematic search for such recurrences.

Contributions For $k = 4, 5, 6, 7$ we improve on the recurrences for Karatsuba-like multiplication. We obtain smaller circuits than the previously known best bounds due to Cenk and Hasan [CH15] for almost all n of cryptographically relevant size. Moreover, we actually construct the circuits (in [CH15] only estimates on the sizes are given). Although we have not focused on depth, our circuits are of significantly lower depth than previous state of the art. For example for $n = 66$ (the smallest value of n for which the interpolation method is used in [Ber09]), our methods yield a circuit with size 4041 and depth 15. The circuit in [Ber09] has size 4050 and depth 79. Table 4 in Section 6 shows circuit sizes and depths for a range of n.

2 Definitions

Polynomials The input to our problem will be polynomials of degree $n-1$,

$$A = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B = \sum_{i=0}^{n-1} b_i x^i \qquad (a_i, b_i \in \{0,1\}).$$

We refer to a polynomial of degree $n-1$ as an n-term polynomial (even though some of the terms may be zero). For integers $i < j$, we identify a polynomial $A = a_i x^i + a_{i+1}x^{i+1} + \ldots + a_j x^j$ with the tuple $(a_i, a_{i+1}, \ldots, a_j)$, and denote $A[i] = a_i$ and $A[i..j] = (a_i, \ldots, a_j)$.

Symmetric Circuits For the purpose of this paper, we restrict ourselves to circuits with the following structure, which we call *symmetric bilinear circuits*. A symmetric bilinear circuit contains only binary XOR (addition) and binary AND (multiplication) gates. It consists of

– a top layer consisting only of XOR gates[1];
– a multiplication layer that computes only functions of the form

$$\sum_{i\in S} a_i \cdot \sum_{i\in S} b_i \qquad (S \subseteq \{0, \ldots, n-1\});$$

– a bottom layer that uses only XOR gates and outputs c_0 through c_{2n-1}, where

$$c_t = \sum_{i+j=t} a_i b_j.$$

For an integer $n > 1$, let $M(n)$ be the size of the smallest circuit over $(\wedge, \oplus, 1)$ computing the polynomial product of two n-term polynomials. We emphasize that there does not necessarily exist a *symmetric boolean circuit* of size $M(n)$ for all n, although all the best sizes we know of can be achieved using circuits of this form.

We will use three metrics for this class of circuits. Let $\mathcal{C}$ be a circuit in the class. The *multiplicative cost of* $\mathcal{C}$, denoted $M_\wedge(\mathcal{C})$, is the number of AND gates. The *upper additive cost of* $\mathcal{C}$, denoted $M^\oplus(\mathcal{C})$, is the number of XOR gates in the top layer of $\mathcal{C}$. The *bottom additive cost of* $\mathcal{C}$, denoted $M_\oplus(\mathcal{C})$, is the number of XOR gates in the bottom layer of $\mathcal{C}$.

Example: Consider $A = a_0 + a_1 x$ and $B = b_0 + b_1 x$. The product of A and B is $C = c_0 + c_1 x + c_2 x^2$, where

$$c_0 = a_0 b_0; c_1 = a_0 b_1 + a_1 b_0; c_2 = a_1 b_1.$$

With respect to multiplicative complexity, there is only one optimal symmetric boolean circuit for $n = 2$. The top layer calculates $s_1 = a_0 + a_1$, $s_2 = b_0 + b_1$. The multiplication layer calculates

[1] We visualize circuits as having the inputs at the top and the outputs at the bottom.

– $c_0 = a_0 b_0$;
– $c_2 = a_1 b_1$;
– $t = s_1 s_2 = (a_0 + a_1)(b_0 + b_1)$.

The bottom layer calculates $c_1 = c_0 + c_2 + t$. The multiplicative cost is three (which is optimal, among all boolean circuits). Both additive costs $M^{\oplus}$ and $M_{\oplus}$ are 2.

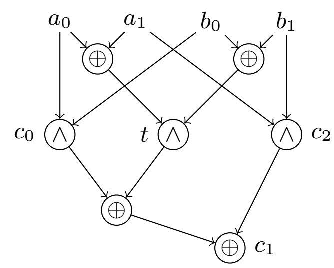

Fig. 1. Circuit $\mathcal{C}$ computing the product of two polynomials of degree 1. This circuit has $M_{\wedge}(\mathcal{C}) = 3$ and $M_{\oplus}(\mathcal{C}) = M^{\oplus}(\mathcal{C}) = 2$.

For degree-2 polynomials, a computer search yields exactly six circuits of multiplicative complexity six in the prescribed class.[2] Of these, two have bottom additive cost 6 (the other four circuits have bottom additive costs 7,7,8,8).

Linear Operators and Representation of Circuits Let A be an n x m matrix over $\mathbb{F}_2$. The function $\mathbf{x} \mapsto A \cdot \mathbf{x}$ can be computed using only XOR gates. We let $s(A)$ be the smallest number of XOR gates in a circuit consisting only of XOR gates computing this mapping. For a symmetric bilinear circuit $\mathcal{C}$ in the variables $\mathbf{a} = (a_0, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, \ldots, b_{n-1})$, there exists a unique matrix T, such that the ith AND gate computes the ith coordinate of $(T \cdot \mathbf{a})^{\top}(T \cdot \mathbf{b})$. We call this matrix T the *top matrix* of $\mathcal{C}$. Similarly, the bottom part of the circuit can be described as a matrix, which we call the *main matrix* of $\mathcal{C}$, denoted R. A symmetric bilinear circuit $\mathcal{C}$ is completely described by the two matrices (T, R) along with XOR circuits computing them.

3 Previous Work

Asymptotic Complexity Much work has been put into giving asymptotically good algorithms for binary polynomial multiplication, see [KO63, Sch77, HvdHL14]. Currently, the asymptotically best algorithm is due to Harvey, van der Hoeven, and Lecerf who showed that $M(n) \leq O(n \log n \; 8^{\log^{\star} n})$, where $\log^{\star}(\cdot)$ denotes the iterated logarithm, an unbounded but extremely slow growing function.

[2] Any symmetric bilinear circuit computing the product of two quadratic polynomials has at least six $\mathbb{F}_2$ multiplications. See Section 5.

Concrete Complexity For values of n that are interesting for cryptographic purposes (say, $n \leq 600$), the asymptotic bounds do not say much about the *concrete circuit complexity*. For this we need to employ a combination of different recursive relations.

This problem has received much attention in recent years, see [Paa96, RHK03, Sun04, Mon05, EYK06, vzGS06a, WP06, Zim07, FH07, PL07, Bod07, BGTZ08, CKO09, FSGL10, Ber09, ZM10, ZMH10, DLV11, CH15].

Bernstein, in [Ber09], used various (new and old) recursive constructions to build small circuits for n-term polynomial multiplication for $2 \leq n \leq 1000$. Notably he obtains results "better than anything that can be found in the hardware literature" [Ber09, page 7], including results reported in [PL07, CKPL05, RHK03, vzGS06b, FSGL10].

In recent work [CH15], Cenk and Hasan show that smaller circuits exist for many values of n. These values were found by finding new recurrence relations, improving on existing recurrence relations, and applying these in a manner similar to that of [Ber09]. Recurrence relations sufficient to obtain the values stated in [CH15] are shown on Table 1.

3.1 Known Recursive Constructions

Many different recursive constructions for polynomial multiplications have been suggested. Most of these constructions are based on one of two ideas:

Interpolation based algorithms Here, to multiply two kn-term polynomials, consider both polynomials as being polynomials of degree $k-1$ with n-term polynomials as coefficients. Then evaluate the polynomials at $2k-1$ points, and perform pointwise multiplications recursively. Finally, obtain the resulting polynomial using interpolation. This general approach was suggested by Toom in [Too63]. Concrete constructions for $k = 2, 3$ have been proposed by Bernstein [Ber09] and by Cenk and Hasan [CH15] (specifically, equations 6,7,9, and 15). These are included in Table 1.

Karatsuba-like algorithms The main observation is that the recursive step in the algorithm for Karatsuba multiplication [KO63] is similar to a particular way of multiplying two 2-term polynomials. Conversely, Karatsuba's algorithm uses a circuit for 2-term multiplication with few multiplications as a recursive way of multiplying $2n$-term polynomials. A generalization of this is to use a particular circuit for multiplication of k-term polynomials as a recurrence for multiplying kn-term polynomials. This has been observed many times, notably by Montgomery [Mon05] and by Weimerskirch and Paar [WP06]. Such recursive constructions have also been proposed in [CHN14, Ber09] and [CH15] (equations 7,18,20), and are listed in Table 1.

We note that this distinction is not a perfect demarcation. In fact, Algorithm 1 on Table 1 (the best known version of the classic Karatsuba algorithm) is presented as an interpolation-based algorithm in [Ber09]. However, as we shall argue in Section 4.2, one can just as well consider it a "Karatsuba-like" algorithm.

Other Recurrences Many other recurrence relations can be found in the literature. For example, Dyka et al. [DLV11] report the recurrences $M(5n) \leq 13M(n)+66n-23$, $M(6n) \leq 17M(n)+96n-34$, $M(7n) \leq 22M(n)+133n-47$. Some recurrences have been patented, as in Montgomery's patent [Mon08] and in Koç and Erdem's patent [KE08]. Our general method for binary polynomial multiplication is described in section 4. It improves on all patented Karatsuba-like recurrences we are aware of.

4 Finding new Karatsuba-like recurrences

This section describes our main technique for obtaining new recurrences. In section 4.1 we introduce Karatsuba-like algorithms and recall a generic way of transforming a circuit into a recursive construction. We then point out why this generic technique is suboptimal. In Section 4.2 we illustrate how to improve on this in the particular case of $k = 2$. In Section 4.3 we consider the case for general values of k.

4.1 Generic Karatsuba-like constructions

There is a standard way to convert any symmetric bilinear circuit $\mathcal{C}$, for polynomial multiplication of k bits, into a recurrence. The recurrence yields an upper bound on $M(k \cdot n)$ in the following way: to multiply two $k \cdot n$-term polynomials A, B, divide A, B into k blocks. That is, write A as

$$A = A_0 + A_1 x^n + \ldots + A_{k-2} x^{n \cdot (k-2)} + A_{k-1} x^{n \cdot (k-1)},$$

with each A_i an n-term polynomial, and similarly write B as

$$B = B_0 + B_1 x^n + \ldots + B_{k-2} x^{n \cdot (k-2)} + B_{k-1} x^{n \cdot (k-1)}.$$

The product $A \cdot B$ can be written as

$$A \cdot B = \sum_{i=0}^{2k-2} U_i x^{i \cdot n},$$

where $U_k = \sum_{i+j=k} A_i \cdot B_j$. To compute the polynomials $U_0, \ldots, U_{2k-2}$ we use the circuit $\mathcal{C}$, where each XOR gate is replaced with polynomial addition (bitwise XOR) and each AND gate is replaced with a circuit for n-term multiplication. Each of the top XOR gates in $\mathcal{C}$ results in n gates, and each of the bottom XOR gates result in $2n - 1$ gates. Each AND gate is replaced by a circuit for n-term multiplication, using $M(n)$ gates. We immediately have that the cost of computing $U_0, \ldots, U_{2k-2}$ is $n \cdot M^{\oplus}(\mathcal{C}) + (2n-1)M_{\oplus}(\mathcal{C}) + M_{\wedge}(\mathcal{C}) \cdot M(n)$. Finally, to obtain the actual bits of the result, we have to take care of the overlap between $U_i x^{i \cdot n}$ and $U_{i+1} x^{(i+1) \cdot n}$. One way to do this is by doing bitwise XOR with the high-order $n - 1$ bits from U_i and the low-order n bits from U_{i+1} for $i = 0, \ldots, 2k - 3$. This uses $(2k - 2) \cdot (n - 1)$ gates.

Table 1. List of known recurrence relations. K/I denotes whether it is a Karatsuba-like or interpolation based algorithm, alg denotes a unique algorithm number (used later when reporting how circuits are obtained), CH is the number of the algorithm used in [CHN14]. Several recurrences are different from what is stated in [CH15]. We have been informed of these improvements by the authors. Most of these recurrences are included in [CH15].

K/I	alg	CH	k	s	$M(s) \leq$	reference
-	0	1	-	$n+1$	$M(n)+4n$	Schoolbook
K	1	2	2	$2n$	$3M(n)+7n-3$	[Ber09]
K	2	2.1	2	$2n-1$	$2M(n)+M(n-1)+7n-8$	[CH15, Eq (3)]
K	N/A	2.2	2	$2n-2$	$2M(n)+M(n-2)+7n-16$	[CH15, Eq (3)]
K	5	3	3	$3n$	$6M(n)+18n-6$	[CHN14]
I	10	5	3	$3n$	$3M(n)+2M(n+2)+35n-21$	[Ber09, CH15]
I	10	5	3	$3n-1$	$2M(n)+M(n-1)+2M(n+1)+35n-26$	[CH15, Eq 6]
I	10	5	3	$3n-2$	$2M(n)+M(n-2)+2M(n+1)+35n-36$	[CH15, Eq 6]
I	N/A	5	3	$3n-3$	$2M(n)+M(n-3)+2M(n+1)+35n-46$	[CH15, Eq 6]
I	N/A	5	3	$3n-4$	$2M(n)+M(n-4)+2M(n+1)+35n-56$	[CH15, Eq 6]
I	11	5.1	3	$3n$	$M(n)+2M(n+1)+M(n+2)+M(n-1)+35n-12$	[CH15, Eq 15]
I	12	5.2	3	$3n-2$	$2M(n)+M(n+1)+2M(n-1)+35n-13$	[CH15, Eq 9]
K	3	6	4	$4n$	$M(2n)+6M(n)+27n-8$	[Ber09], [CH15, Eq 9]
K	3	6.1	4	$4n-1$	$M(2n)+5M(n)+M(n-1)+27n-18$	[Ber09], [CH15, Eq 10]
K	3	6.2	4	$4n-2$	$M(2n)+5M(n)+M(n-2)+27n-34$	[Ber09], [CH15, Eq 10]
K	7	7	5	$5n$	$13M(n)+55n-17$	[CH15, Eq 18]
K	7	7.1	5	$5n-1$	$12M(n)+M(n-1)+55n-26$	[CH15, Eq 20]

This gives the generic recurrence

$$M(kn) \leq M_{\wedge}(\mathcal{C})M(n) + M^{\oplus}(\mathcal{C})n + M_{\oplus}(\mathcal{C})(2n-1) + (n-1)(2k-2). \quad (1)$$

This idea has been used before, though we have not seen the above recurrence stated explicitly.

The Bottom Layer To improve on this, we "zoom in" on how to obtain the output bits given the result of the recursive multiplications. Informally speaking, we lose information by obtaining the polynomials U_i and the overlap as independent tasks. This general idea appeared (somewhat independently) in several works [Mon05, ZM10, DLV11, Nèg14, CHN14].

4.2 Example: Karatsuba ($k = 2$)

To illustrate how to optimize the bottom layer, we describe a way to obtain the Karatsuba-recurrence $M(2n) \leq 3M(n) + 7n - 3$ in table 2. This recurrence is not novel, but this particular way of deriving it naturally generalizes to the more general approach we present in the next section. Let the input polynomials be

$$A = a_0 + a_1x + \ldots + a_{2n-1}x^{2n-1} = A_0 + A_1x^n,$$

and

$$B = b_0 + b_1x + \ldots + b_{2n-1}x^{2n-1} = B_0 + B_1x^n,$$

for n-term polynomials A_0, A_1, B_0, B_1. Let the result be

$$C = A \cdot B = c_0 + c_1x + \ldots + c_{4n-2}x^{4n-2},$$

and let $U_k = \sum_{i+j=k} A_i \cdot B_j$, for $k = 0, 1, 2$. Now instantiate Equation 1 with the circuit described by the two matrices (also shown on Figure 2)

$$T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, R = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

The circuit is shown on Figure 1. We get

$$M(2n) \leq 3M(n) + 2n + 2 \cdot (2n-1) + (n-1)(2 \cdot 2 - 2) = 3M(n) + 8n - 4.$$

To improve on this, we consider the overlap more carefully. For a $(2n-1)$-term polynomial D let $L(D)$, $M(D)$, and $H(D)$ be the unique polynomials with $n-1$, 1, and $n-1$ terms, respectively, that satisfy

$$D = L(D) + M(D)x^{n-1} + H(D)x^n.$$

Consider the recursion circuit as shown in Figure 2. Different parts of the output C can be written in terms of the coefficients T_0, T_1, T_2. Using the fact that $U_0 = P_0, U_1 = P_0 + P_1 + P_2$, $U_2 = P_2$, we can write the output bits as

1. $C[0..n-2] = L(U_0) = L(P_0)$,
2. $C[n-1] = M(U_0) = M(P_0)$,
3. $C[n..2n-2] = L(U_1) + H(U_0) = (L(P_0) + L(P_1) + L(P_2)) + H(P_0)$,
4. $C[2n-1] = M(U_1) = M(P_0) + M(P_1) + M(P_2)$,
5. $C[2n..3n-2] = L(U_2) + H(U_1) = L(P_2) + (H(P_0) + H(P_1) + H(P_2))$,
6. $C[3n-1] = M(U_2) = M(P_2)$,
7. $C[3n..4n-2] = H(U_2) = H(P_2)$.

Now we can write the outputs as linear functions of the low, middle, and high parts of the polynomials computed in the multiplication gates:

$$\begin{pmatrix} C[n-1] \\ C[2n-1] \\ C[3n-1] \end{pmatrix} = \begin{pmatrix} 1\ 0\ 0 \\ 1\ 1\ 1 \\ 0\ 0\ 1 \end{pmatrix} \cdot \begin{pmatrix} M(P_0) \\ M(P_1) \\ M(P_2) \end{pmatrix}, \tag{2}$$

and

$$\begin{pmatrix} C[0..n-2] \\ C[n..2n-2] \\ C[2n..3n-2] \\ C[3n..4n-2] \end{pmatrix} = \begin{pmatrix} 1\ 0\ 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ 1\ 0\ 0 \\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 1 \end{pmatrix} \cdot \begin{pmatrix} L(P_0) \\ L(P_1) \\ L(P_2) \\ H(P_0) \\ H(P_1) \\ H(P_2) \end{pmatrix}. \tag{3}$$

Now it remains to find good circuits to compute the linear operators in Equations 2 and 3. Since these matrices are small, it is easy to see that the first can be computed using two additions, and the second can be computed using five additions. Each addition in the computation of the first linear mapping costs 1 XOR gate, and in second linear mapping each operation costs $n-1$ XOR operations. The top part still uses $2n$ XOR gates. We have in total

$$M(2n) \leq 3M(n) + 2n + 5 \cdot (n-1) + 2 = 3M(n) + 7n - 3. \tag{4}$$

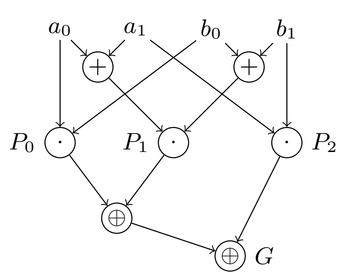

Fig. 2. Circuit showing the recursion in the Karatsuba step. $U_0 = P_1, U_1 = G, U_2 = P_2$.

4.3 Generalizing to $k \geq 3$

Let $\mathcal{C}$ be a symmetric bilinear boolean circuit for multiplication of k-term polynomials with top matrix T and main matrix R. Let R_i be the ith row of R. Consider $\mathcal{C}$ as a recursive circuit (as in the previous section). Let $P_0, \ldots, P_{s-1}$ be the multiplication gates. The output polynomial C satisfies

$$\begin{pmatrix} C[n-1] \\ C[2n-1] \\ \vdots \\ C[(2k-1)n-1] \end{pmatrix} = R \cdot \begin{pmatrix} M(P_0) \\ M(P_1) \\ \vdots \\ M(P_{s-1}) \end{pmatrix}. \tag{5}$$

Let the *extended matrix* E be defined as

$$E = \begin{pmatrix} R_1 & 0 \\ R_2 & R_1 \\ \vdots & \\ R_{2k-1} & R_{2k-2} \\ 0 & R_{2k-1} \end{pmatrix}.$$

Letting L, M, H be as in the previous section, the remaining coefficients can be written as

$$\begin{pmatrix} C[0..n-2] \\ C[n..2n-2] \\ \vdots \\ C[(2k-1)n..2kn-2] \end{pmatrix} = E \cdot \begin{pmatrix} L(P_0) \\ \vdots \\ L(P_{s-1}) \\ H(P_0) \\ \vdots \\ H(P_{s-1}) \end{pmatrix}. \tag{6}$$

We now have the recurrence

$$M(kn) \leq M_\wedge(\mathcal{C}) \cdot M(n) + 2n \cdot s(T) + (n-1) \cdot s(E) + s(R). \tag{7}$$

Note that this allows for a succinct description of recursive circuit: each is described by XOR circuits for T, R and E.

Matrix computation vs using common subexpressions For specific values of k, similar approaches have been used. To quote Montgomery [Mon05, p. 365]: "by taking advantage of common subexpressions", and Zhou and Michalik [ZM10]: "By reviewing the work of [Paa96][3], we show that the gate complexity of KA can be reduced by exploring the common subexpressions".

We remark that obtaining a circuit for linear operators purely using common subexpressions results in so-called "cancellation-free" circuits (also called SUM-circuits). For some linear operators these circuits are highly suboptimal [BF15], see also [JS13, Section 5.3]. Indeed, for the extended matrix E in the 6-way split below, the only minimal sized circuit we have found has cancellation, so it could not have been obtained using only common subexpressions.

[3] reference name changed to be consistent with citations in this document

Finding Recursion Circuits The recurrence in Equation 7 suggests the following strategy to finding recurrences upper bounding $M(k \cdot n)$ for a fixed k: First find circuits for k-term multiplication with the smallest possible number of AND gates. Among these, find one where Equation 7 is as good as possible.

We remark that both of these tasks are computationally very challenging; computing the tensor rank is **NP**-hard [Hås90]. The problem of finding the smallest XOR circuit for a given matrix is **NP**-hard and max-**SNP**-hard [BMP13], meaning that if $\mathbf{NP} \neq \mathbf{P}$ even finding a circuit which is at most a particular constant larger than the optimum is intractable. For this work we used the heuristics of [BMP13].

4.4 New Recurrence Relations

Using the approach described in the previous sections we obtain several new recurrence relations. These are shown on Table 2. We describe the recurrence by describing the two matrices T, R associated with the recurrence (the matrix E is derived from R). The straight-line programs computing each of the matrices are given in the appendix. We only include the recurrences in the case where the input is divisible by $4, 5, 6, 7$, although these can easily be extended to give upper bounds for $M(4n-1), M(4n-2), M(5n-1)$, etc. We omit this in this version of the paper, though these algorithms are included in the software computing our circuits.

Table 2. Recurrence relation for new Karatsuba-like algorithms

alg	recurrence	reference
6	$M(4n) \leq 9M(n) + 34n - 12$	Eq 8
7	$M(5n) \leq 13M(n) + 54n - 19$	Eq 9
8	$M(6n) \leq 17M(n) + 85n - 29$	Eq 10
9	$M(7n) \leq 22M(n) + 107n - 33$	Eq 11

3-way split A search gives the matrices

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

It is easily seen that $s(T) = 3$, $s(R) \leq 6$, and $s(E) \leq 12$. This gives the recurrence

$$M(3n) \leq 6M(n) + 2 \cdot 3n + (n-1)12 + 6 = 6M(n) + 18n - 6,$$

which is the same recurrence as reported in [CHN14].

4-way split A computer search gives the matrices

$$T = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 1&1&0&0 \\ 0&0&1&0 \\ 1&0&1&0 \\ 0&0&0&1 \\ 0&1&0&1 \\ 0&0&1&1 \\ 1&1&1&1 \end{pmatrix}, R = \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0 \\ 1&1&0&1&1&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1 \\ 0&1&0&1&0&1&1&0&0 \\ 0&0&0&1&0&1&0&1&0 \\ 0&0&0&0&0&1&0&0&0 \end{pmatrix},$$

again it is not hard to verify that $s(T) \leq 5$, $s(R) \leq 12, s(E) \leq 24$, and that this uses 9 multiplications. We get the recurrence

$$M(4n) \leq 9M(n) + 2 \cdot 5n + (n-1)24 + 12 = 9M(n) + 34n - 12. \tag{8}$$

We note that this is a little better than what one would get by applying Equation 4 twice.

5-way split For $n = 5$, a computer search gave matrices T, M, E with $s(T) \leq 8$, $s(R) \leq 19$, and $S(E) \leq 38$, using 13 multiplications. The matrices along with straight-line programs are given in Section A.1. This gives the recurrence

$$M(5n) \leq 13M(n) + 2 \cdot 8n + 38(n-1) + 19 = 13M(n) + 54n - 19. \tag{9}$$

6-way split For $n = 6$, a computer search found matrices T, M, E with $s(T) \leq 13$ gates, $s(R) \leq 30$, and $s(E) \leq 59$, using 17 multiplications. The matrices along with straight-line programs are given in Section A.2 This gives the recurrence

$$T(6n) \leq 17M(n) + 2 \cdot 13n + (n-1) \cdot 59 + 30 = 17M(n) + 85n - 29. \tag{10}$$

7-way split Similarly for $n = 7$ a computer search found matrices T, M, E with $s(T) \leq 16$, $s(R) \leq 41$, and that $s(E) \leq 75$ gates, using 22 multiplications. The matrices along with straight-line programs are given in Section A.3 This leads to the recurrence

$$M(7n) \leq 22M(n) + 2 \cdot 16n + 75(n-1) + 41 = 22M(n) + 107n - 34. \tag{11}$$

5 Multiplicative Complexity of Polynomial Multiplication

A natural question about the recurrences in the previous section is whether they can be improved; Do matrices giving better recurrence relations exist? In particular, is it possible to find matrices giving a smaller number of multiplications in the recursion. It turns out that using this technique, for $k = 2, 3, 4, 5, 6$ there is not, but for $k \geq 7$ there could be. In this section we will briefly sketch why.

There is a known relationship between error correcting codes and quadratic boolean circuits computing finite field or polynomial multiplication (see [BD80], [LSW83]). Roughly speaking, any quadratic boolean circuit computing n-term polynomial multiplication induces an error correcting code with certain parameters. Therefore the nonexistence of certain codes can be used to prove the nonexistence of certain circuits. More specifically, Kaminski [Kam85] shows that if there exists a method for multiplying two n-term polynomials in $\mathbb{F}_2[X]$ which uses l multiplications over $\mathbb{F}_2$, then there exists a linear code of length l, weight $2n - d$, and dimension d. This holds for all $n \leq d < 2n$. Table 3 gives multiplicative complexity lower bounds derived using known bounds for the length of linear codes (see e.g. [Gra07]). We leave it as open problems to close the gap between 20 and 22 for $k = 7$, and to find recurrences with better low-order terms than what we provide in the previous section.

Table 3. Best upper and lower bound on the multiplicative complexity of polynomial multiplication. The column (l, d, w) indicates parameters for a code with length l, dimension d and weight w that does not exist and therefore establishes the lower bound.

n	lower	(l, d, w)	upper
2	3	$(2, 2, 2)$	3
3	6	$(5, 3, 3)$	6
4	9	$(8, 3, 5)$	9
5	13	$(12, 5, 5)$	13
6	17	$(16, 3, 9)$	17
7	20	$(19, 5, 9)$	22

6 Results

We use the technique of section 4 to obtain circuits inductively: first find small circuits for $n = 2, 3 \ldots, k$. Then, to find a circuit for multiplication of $(k + 1)$-term polynomials, apply each of the applicable recursive constructions, using the previously found circuits as base cases. Then look at obvious inefficiencies (unused gates, two distinct gates computing the same function, etc.). Finally, select the smallest circuit and continue. Table 4 shows the obtained circuit sizes along with their depths. Since the depths of circuits that use interpolation may be too large for practical applications, we also include the sizes and depths of circuits that do not use interpolation. All circuits for $n = 2, 3, \ldots, 109$ and $n = 135, 136, 137, 189, 191, 233, 283$ are attached in this submission. As an example, in Section B, we include a circuit for $n = 15$. Note for even this small value we find a circuit using five fewer gates and has a depth 3 smaller (reducing with 1.5% and 18.75%, respectively).

Table 4: Circuit sizes and depths. s_{CH}, d_{CH} refer to the sizes and depths reported in [CH15].
Alg denotes what recurrence is used, the number of the algorith, following the numbers given on Table 1 and 2.
The * for $n = 11$ indicates that the circuit published in [Per14] is used.

	all algorithms					Karatsuba algorithms only				
n	s_{CH}	s	d_{CH}	d	alg.	s_{CH}	s	d_{CH}	s_d	alg.
2	5	5	2	2	(0)	5	5	2	2	(0)
3	13	13	3	3	(0)	13	13	3	3	(0)
4	25	25	4	4	(0)	25	25	4	4	(0)
5	41	41	5	5	(0)	41	41	5	5	(0)
6	57	57	6	6	(1)	57	57	6	6	(1)
7	81	81	7	7	(0)	81	81	7	7	(0)
8	100	100	7	7	(1)	100	100	7	7	(1)
9	126	126	7	7	(5)	126	126	7	7	(5)
10	155	**154**	8	8	(7)	155	**154**	8	8	(7)
11	186	186	7	7	*	186	186	7	7	*
12	207	207	7	8	6	207	207	7	8	6
13	255	255	8	9	0	255	255	8	9	0
14	289	289	10	**9**	1	289	289	10	**9**	1
15	317	**312**	16	**13**	7	317	**312**	16	**13**	7
16	349	349	8	9	6	349	349	8	9	6
17	407	**406**	10	**9**	2	407	**406**	10	**9**	2
18	438	438	10	**9**	1	438	438	10	**9**	1
19	498	**495**	11	15	2	498	**495**	11	15	2
20	527	**522**	8	14	7	527	**522**	8	14	7
21	596	**573**	11	14	9	596	**573**	11	14	9
22	632	632	10	10	1	632	632	10	10	1
23	676	**675**	10	11	2	676	**675**	10	11	2
24	702	702	10	11	1	702	702	10	11	1
25	791	**784**	18	**14**	7	791	**784**	18	**14**	7
26	853	853	11	12	1	853	853	11	12	1
27	912	912	11	**10**	5	912	912	11	**10**	5
28	956	**944**	15	15	9	956	**944**	15	15	9
29	1020	**1009**	19	**16**	2	1020	**1009**	19	**16**	2
30	1053	**1038**	19	**16**	1	1053	**1038**	19	**16**	1
31	1119	**1113**	19	**15**	2	1119	**1113**	19	**15**	2
32	1156	1156	11	12	1	1156	1156	11	12	1
33	1274	**1271**	13	**12**	2	1274	**1271**	13	**12**	2
34	1335	**1333**	13	**12**	4	1335	**1333**	13	**12**	4
35	1393	**1392**	15	**11**	3	1393	**1392**	15	**11**	3
36	1429	**1428**	15	**12**	6	1429	**1428**	15	**12**	6
37	1559	**1552**	14	18	2	1559	**1552**	14	18	2

38	1616	**1604**	13	16	4	1616	**1604**	13	16	4
39	1680	**1669**	13	17	3	1680	**1669**	13	17	3
40	1718	**1703**	11	17	1	1718	**1703**	11	17	1
41	1858	**1806**	14	17	2	1858	**1806**	14	17	2
42	1929	**1862**	13	17	9	1929	**1862**	13	17	9
43	1996	**1982**	15	16	2	1996	**1982**	15	16	2
44	2037	**2036**	15	**13**	6	2037	**2036**	15	**13**	6
45	2116	**2105**	20	**17**	7	2116	**2105**	20	**17**	7
46	2182	**2179**	15	17	3	2182	**2179**	15	17	3
47	2229	**2228**	15	**13**	3	2229	**2228**	15	**13**	3
48	2260	**2259**	15	**14**	6	2260	**2259**	15	**14**	6
49	2451	**2436**	21	**17**	2	2451	**2436**	21	**17**	2
50	2545	**2523**	21	23	7	2545	**2523**	21	23	7
51	2668	**2663**	16	16	2	2668	**2663**	16	16	2
52	2726	**2725**	16	**15**	6	2726	**2725**	16	**15**	6
53	2858	**2841**	14	24	8	2858	**2841**	14	24	8
54	2922	**2878**	14	24	8	2922	**2878**	14	24	8
55	3006	**2987**	20	**18**	2	3006	**2987**	20	**18**	2
56	3060	**3022**	20	**18**	9	3060	**3022**	20	**18**	9
57	3191	**3145**	22	**18**	3	3191	**3145**	22	**18**	3
58	3256	**3212**	22	**19**	4	3256	**3212**	22	**19**	4
59	3304	**3273**	20	**18**	3	3304	**3273**	20	**18**	3
60	3334	**3306**	20	**19**	6	3334	**3306**	20	**19**	6
61	3500	**3472**	22	**18**	2	3500	**3472**	22	**18**	2
62	3571	**3553**	22	**18**	1	3571	**3553**	22	**18**	1
63	3632	**3626**	21	**18**	3	3632	**3626**	21	**18**	3
64	3674	**3673**	16	**15**	6	3674	**3673**	16	**15**	6
65	3927	**3919**	16	45	10	3927	**3920**	16	**15**	2
66	4040	**3998**	86	**45**	11	4048	**4041**	16	**15**	1
67	4110	**4075**	88	**45**	12	4159	**4152**	18	**14**	3
68	4167	**4153**	88	**45**	10	4228	**4220**	18	**14**	3
69	4296	**4271**	97	**47**	11	4356	**4353**	18	**14**	2
70	4374	**4332**	99	**47**	12	4420	**4417**	20	**14**	3
71	4476	**4449**	99	**47**	10	4494	**4478**	20	26	8
72	4535	**4510**	99	**26**	8	4535	**4510**	20	26	8
73	4701	**4654**	20	48	12	4798	**4781**	18	24	7
74	4839	**4813**	101	**24**	7	4892	**4813**	29	**24**	7
75	4929	**4847**	101	**24**	7	4929	**4847**	29	**24**	7
76	5097	**5050**	29	51	12	5109	**5075**	18	19	1
77	5205	**5186**	103	**51**	10	5241	**5198**	16	19	3
78	5297	**5255**	101	**19**	3	5297	**5255**	16	19	3
79	5359	**5329**	16	20	3	5359	**5329**	29	**20**	3
80	5400	**5366**	29	**20**	6	5400	**5366**	21	**20**	6
81	5630	**5578**	21	56	11	5713	**5593**	17	20	2
82	5723	**5655**	110	**56**	12	5854	**5702**	16	20	1

83	5818	**5760**	112	**56**	10	5983	**5769**	18	20	9
84	5929	**5804**	112	**20**	9	6064	**5804**	18	20	9
85	6007	**5913**	11	56	12	6209	**6118**	23	**19**	2
86	6091	**6015**	115	**56**	10	6284	**6224**	20	20	4
87	6204	**6128**	115	**57**	11	6369	**6344**	20	**19**	3
88	6302	**6210**	116	**57**	12	6415	**6413**	20	**16**	1
89	6388	**6322**	118	**57**	10	6576	**6516**	23	29	8
90	6500	**6443**	118	**58**	10	6660	**6550**	23	29	8
91	6572	**6497**	117	**57**	12	6794	**6776**	23	**20**	2
92	6662	**6623**	120	**57**	10	6851	**6842**	20	**19**	3
93	6831	**6790**	120	**60**	11	6944	**6929**	23	**19**	3
94	6931	**6883**	120	**60**	12	7013	**7010**	18	**16**	1
95	7073	**7049**	122	**60**	10	7076	**7073**	20	**17**	2
96	7112	**7110**	120	**17**	1	7112	**7110**	20	**17**	1
97	7337	**7296**	20	60	12	7496	**7465**	21	**20**	2
98	7503	**7481**	121	**60**	10	7684	**7636**	24	25	4
99	7636	**7611**	121	**63**	11	7859	**7801**	26	**25**	3
100	7766	**7740**	124	**63**	10	7934	**7847**	21	25	7
101	7894	**7873**	126	**63**	10	8230	**8197**	24	27	2
102	7979	**7977**	126	**63**	11	8345	**8318**	24	27	8
103	8097	**8057**	129	**63**	12	8466	**8361**	23	25	9
104	8178	**8160**	129	**63**	10	8538	**8398**	21	25	9
105	8358	**8329**	129	**70**	11	8805	**8435**	19	25	9
106	8450	**8406**	129	**70**	12	8932	**8861**	19	27	8
107	8603	**8574**	131	**70**	10	8998	**8904**	31	**27**	8
108	8758	**8719**	131	**67**	10	9040	**8947**	31	**27**	8
109	8874	**8813**	131	**67**	12	9311	**9221**	23	**20**	3
135	12453	**12273**	163	**81**	11	13077	**12988**	23	47	3
136	12422	**12360**	165	**81**	12	13148	**13061**	23	47	3
137	12522	**12491**	163	**81**	10	13415	**13332**	21	49	3
189	20671	**20621**	218	**108**	11	21766	**21745**	25	**22**	3
191	21048	**21014**	218	**108**	10	21919	**21910**	25	**19**	3
233	29156	**29058**	274	**129**	10	31381	**31365**	43	**26**	7
283	38432	38555	414	**153**	12	42468	**42316**	45	53	6

7 Conclusion

In this work we proposed a new way to describe, find, and analyze Karatsuba-like recurrences, and found better recurrences than previously known for splitting into $4, 5, 6$, and 7 blocks. Using these recurrences together with known recurrences we constructed circuits for binary polynomial multiplication better than previously known. These circuits may be used as components in software for cryptographic purposes, such at batch evaluations of finite field multiplications or multiparty computation. They may also be used as a basis for a hardware implementation of polynomial or finite field multiplication. To do this one would take a particular circuit and do additional optimizations to accommodate practical constraints, and maybe use additional techniques to decrease size or depth. The circuits were verified by computing the algebraic normal form of the outputs.

The software used in this project is flexible in terms of evaluation criteria. In this work we have focused on size, but future work includes focusing on the number of multiplications, area or energy optimization, as well as other criteria. Additional future work is to apply the ideas used in this work to other operations, such as matrix multiplication and finite field multiplication.

We point out that the circuits for our new recurrence relations (section 4.4) were computed without using depth as a secondary optimality criteria. This causes the depth of circuits reported here to be much larger than we can actually obtain. For example, for $n = 15$ Table 4 reports a depth of 13. The actual depth we can obtain, without increasing the size, is no bigger than 9. The final version of this paper will contain the recomputed depths and circuits.

References

[BD80] Mark R. Brown and David P. Dobkin. An improved lower bound on polynomial multiplication. *IEEE Trans. Computers*, 29(5):337–340, 1980.

[Ber09] Daniel J. Bernstein. Batch binary edwards. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 317–336. Springer, 2009.

[BF15] Joan Boyar and Magnus Gausdal Find. Cancellation-free circuits in unbounded and bounded depth. *Theor. Comput. Sci.*, 590:17–26, 2015.

[BGTZ08] Richard P. Brent, Pierrick Gaudry, Emmanuel Thomé, and Paul Zimmermann. Faster multiplication in GF(2)[x]. In Alfred J. van der Poorten and Andreas Stein, editors, *Algorithmic Number Theory, 8th International Symposium, ANTS-VIII, Banff, Canada, May 17-22, 2008, Proceedings*, volume 5011 of *Lecture Notes in Computer Science*, pages 153–166. Springer, 2008.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptology*, 26(2):280–312, 2013.

[Bod07] Marco Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In Claude Carlet and Berk Sunar, editors, *WAIFI 2007 proceedings*, volume 4547 of *LNCS*, pages 116–133. Springer, June 2007. `http://bodrato.it/papers/\#WAIFI2007`.

[CH15] Murat Cenk and M. Anwar Hasan. Some new results on binary polynomial multiplication. *IACR Cryptology ePrint Archive*, 2015:94, 2015.

[CHN14] Murat Cenk, M. Anwar Hasan, and Christophe Nègre. Efficient subquadratic space complexity binary polynomial multipliers based on block recombination. *IEEE Trans. Computers*, 63(9):2273–2287, 2014.

[CKO09] Murat Cenk, Cetin Kaya Koç, and Ferruh Ozbudak. Polynomial multiplication over finite fields using field extensions and interpolation. In *2009 19th IEEE International Symposium on Computer Arithmetic*, pages 84–91. IEEE, 2009.

[CKPL05] Nam Su Chang, Chang Han Kim, Young-Ho Park, and Jongin Lim. A non-redundant and efficient architecture for Karatsuba-Ofman algorithm. In *Information Security*, pages 288–299. Springer, 2005.

[DLV11] Zoya Dyka, Peter Langendoerfer, and Frank Vater. Combining multiplication methods with optimized processing sequence for polynomial multiplier in $gf(2^k)$. In *Proceedings of the 4th Western European conference on Research in Cryptology*, pages 137–150. Springer-Verlag, 2011.

[Dwo07] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. *NIST Special Publication*, (SP800-38D), November 2007.

[EYK06] Serdar S Erdem, Tuğrul Yanık, and Çetin K Koç. Polynomial basis multiplication over gf(2^m). *Acta Applicandae Mathematicae*, 93(1):33–55, 2006.

[FH07] Haining Fan and M Anwar Hasan. Comments on "five, six, and seven-term karatsuba-like formulae". *IEEE Trans. Computers*, 56(5):716–717, 2007.

[FSGL10] Haining Fan, Jian Sun, Ming Gu, and Kam-Yiu Lam. Overlap-free Karatsuba-Ofman polynomial multiplication algorithms. *Information Security, IET*, 4(1):8–14, 2010.

[Gra07] Markus Grassl. Bounds on the minimum distance of linear codes and quantum codes. Online available at `http://www.codetables.de`, 2007. Accessed on 2015-09-10.

[Hås90] Johan Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[HvdHL14] David Harvey, Joris van der Hoeven, and Grégoire Lecerf. Faster polynomial multiplication over finite fields. *CoRR*, abs/1407.3361, 2014.

[JS13] Stasys Jukna and Igor Sergeev. Complexity of linear boolean operators. *Foundations and Trends in Theoretical Computer Science*, 9(1):1–123, 2013.

[Kam85] Michael Kaminski. A lower bound for polynomial multiplication. *Theor. Comput. Sci.*, 40:319–322, 1985.

[KE08] Ç.K. Koç and S.S. Erdem. Lean multiplication of multi-precision numbers over gf(2m), November 4 2008. US Patent 7,447,310.

[KO63] Anatoly A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963. Available at: `http://cr.yp.to/bib/entries.html#1963/karatsuba`.

[LSW83] Abraham Lempel, Gadiel Seroussi, and Shmuel Winograd. On the complexity of multiplication in finite fields. *Theor. Comput. Sci.*, 22:285–296, 1983.

[Mon05] Peter L. Montgomery. Five, six, and seven-term Karatsuba-like formulae. *IEEE Trans. Computers*, 54(3):362–369, 2005.

[Mon08] P.L. Montgomery. Six-term karatsuba-variant calculator, April 22 2008. US Patent 7,363,336.

[Nèg14] Christophe Nègre. Efficient binary polynomial multiplication based on optimized karatsuba reconstruction. *J. Cryptographic Engineering*, 4(2):91–106, 2014.

[Paa96] Christof Paar. A new architecture for a parallel finite field multiplier with low complexity based on composite fields. *IEEE Trans. Computers*, 45(7):856–861, 1996.

[Per14] René Peralta. Circuit minimization work, http://cs-www.cs.yale.edu/homes/peralta/circuitstuff/cmt.html, january 2014.

[PL07] Steffen Peter and Peter Langendörfer. An efficient polynomial multiplier in gf (2 m) and its application to ECC designs. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1253–1258. EDA Consortium, 2007.

[RHK03] Francisco Rodrıguez-Henrıquez and Çetin Kaya Koç. On fully parallel Karatsuba multipliers for $GF(2^m)$. In *Proc. International Conference on Computer Science and Technology-CST*, pages 405–410, 2003.

[Sch77] Arnold Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Inf.*, 7:395–398, 1977.

[Sun04] Berk Sunar. A generalized method for constructing subquadratic complexity $GF(2^k)$ multipliers. *Computers, IEEE Transactions on*, 53(9):1097–1105, 2004.

[Too63] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3(4):714–716, 1963.

[vzGS06a] Joachim von zur Gathen and Jamshid Shokrollahi. Efficient FPGA-based karatsuba multipliers for polynomials over $\mathbb{F}_2$. In *Selected Areas in Cryptography*, pages 359–369. Springer, 2006.

[vzGS06b] Joachim von zur Gathen and Jamshid Shokrollahi. Fast arithmetic for polynomials over F_2 in hardware. In *Information Theory Workshop, 2006. ITW'06 Punta del Este. IEEE*, pages 107–111. IEEE, 2006.

[WP06] André Weimerskirch and Christof Paar. Generalizations of the karatsuba algorithm for efficient implementations. *IACR Cryptology ePrint Archive*, 2006:224, 2006.

[Zim07] Paul Zimmermann. How fast can we multiply polynomials over $GF(2)$?[extended abstract]. In *Proceedings of Conference on Algorithmic Number Theory 2007*, page 165, 2007.

[ZM10] Gang Zhou and Harald Michalik. Comments on" a new architecture for a parallel finite field multiplier with low complexity based on composite field". *IEEE Trans. Computers*, 59(7):1007–1008, 2010.

[ZMH10] Gang Zhou, Harald Michalik, and László Hinsenkamp. Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(7):1057–1066, 2010.

A Matrices and Their Straight-Line Programs

A.1 5-way Split

Top matrix The following is a straight-line program computing the top matrix from the 5-way split in Section 4.4.

```
8 gates
5 inputs
a0 a1 a2 a3 a4
13 outputs
baseA0 baseA1 baseA2 baseA3 baseA4 baseA5 baseA6
baseA7 baseA8 baseA9 baseA10 baseA11 baseA12
begin
baseA0 = a0
baseA1 = a1
baseA3 = a2
```

```
baseA5 = a3
baseA7 = a4
X5 = a0 + a1
baseA2 = X5
X6 = a0 + a2
baseA4 = X6
X7 = a2 + a4
baseA8 = X7
X8 = a1 + X7
baseA9 = X8
X9 = a3 + a4
baseA10 = X9
X10 = a3 + X6
baseA6 = X10
X11 = X5 + X9
baseA11 = X11
X12 = a2 + X11
baseA12 = X12
end
```

Main matrix The following is a straight-line program computing the main matrix from the 5-way split in Section 4.4.

```
19 gates
13 inputs
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12
9 outputs
row0 row1 row2 row3 row4 row5 row6 row7 row8
begin
row0 = T0
row8 = T7
X13 = T0 + T1
X14 = T2 + X13
row1 = X14
X15 = T5 + T7
X16 = T10 + X15
row7 = X16
X17 = T3 + T4
X18 = X13 + X17
row2 = X18
X19 = T3 + T8
X20 = X15 + X19
row6 = X20
X21 = T6 + T12
X22 = T0 + T11
X23 = X20 + X21
```

```
X24 = X22 + X23
row3 = X24
X25 = T9 + X14
X26 = X16 + X21
X27 = X25 + X26
row4 = X27
X28 = T7 + T9
X29 = T11 + T12
X30 = X18 + X28
X31 = X29 + X30
row5 = X31
end
```

Extended matrix The following is a straight-line program computing the extended matrix from the 5-way split in Section 4.4.

```
38 gates
26 inputs
u0 u1 u2 u3 u4 u5 u6 u7 u8 u9 u10 u11 u12
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12
8 outputs
row0 row1 row2 row3 row4 row5 row6 row7
begin
X26 = u0 + v0
X27 = u7 + v7
X28 = v5 + X27
X29 = v10 + X28
row7 = X29
X30 = u1 + X26
X31 = u2 + X30
row0 = X31
X32 = u3 + v1
X33 = u5 + v3
X34 = X28 + X33
X35 = u10 + X34
X36 = v8 + X35
row6 = X36
X37 = X30 + X32
X38 = v2 + X37
X39 = u4 + X38
row1 = X39
X40 = u9 + v6
X41 = u12 + v12
X42 = X40 + X41
X43 = X32 + X33
X44 = v4 + X43
```

```
X45 = u8 + X44
X46 = X39 + X42
X47 = u11 + X29
X48 = v9 + X46
X49 = X47 + X48
row4 = X49
X50 = X36 + X42
X51 = X31 + X50
X52 = v11 + X51
X53 = u6 + X52
row3 = X53
X54 = X27 + X45
X55 = v12 + X54
X56 = v0 + v9
X57 = X55 + X56
X58 = v11 + X57
row5 = X58
X59 = X26 + X55
X60 = X41 + X59
X61 = v7 + X60
X62 = u6 + X61
X63 = u11 + X62
row2 = X63
end
```

A.2 6-way Split

The matrices used in the 6-way split are as follows:

$$T = \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 1&1&0&0&0&0\\ 0&1&1&0&0&0\\ 1&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&0&0&1&0\\ 0&1&0&0&1&0\\ 0&0&0&1&1&0\\ 1&1&0&1&1&0\\ 0&0&0&0&0&1\\ 1&0&1&0&0&1\\ 1&0&0&1&0&1\\ 1&0&1&1&0&1\\ 0&0&0&0&1&1\\ 0&1&1&0&1&1\\ 0&0&0&1&1&1 \end{pmatrix},$$

and

$$
R = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&1&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1 \\
0&1&1&0&0&1&0&1&0&1&0&1&0&1&1&0&1 \\
1&1&0&0&0&0&1&1&0&0&1&1&1&1&0&0&0 \\
0&0&1&0&1&1&1&1&0&0&0&0&1&1&1&1&0 \\
0&1&1&1&1&1&1&0&1&0&0&0&1&1&0&0&0 \\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1 \\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0 \\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0
\end{pmatrix}.
$$

Top matrix The following is a straight-line program computing the top matrix from the 6-way split in Section 4.4.

```
13 gates
6 inputs
a0 a1 a2 a3 a4 a5
17 outputs
baseA0 baseA1 baseA6 baseA10 baseA2 baseA3 baseA4 baseA7 baseA5
baseA8 baseA14 baseA16 baseA9 baseA15 baseA13 baseA12 baseA11
begin
baseA0 = a0
baseA1 = a1
baseA6 = a4
baseA10 = a5
baseA2 = baseA0 + baseA1
baseA3 = baseA1 + a2
baseA4 = baseA0 + baseA3
baseA7 = baseA1 + baseA6
baseA5 = a2 + a3
baseA8 = a3 + baseA6
baseA14 = baseA6 + baseA10
baseA16 = a3 + baseA14
baseA9 = baseA2 + baseA8
baseA15 = baseA3 + baseA14
baseA13 = baseA9 + baseA15
baseA12 = a2 + baseA13
baseA11 = a3 + baseA13
end
```

Main matrix The following is a straight-line program computing the main matrix from the 7-way split in Section 4.4.

```
30 gates
```

```
17 inputs
u0 u1 u2 u3 u4 u5 u6 u7 u8 u9 u10 u11 u12 u13 u14 u15 u16
11 outputs
row0 row10 row8 row2 row7 row3 row9 row1 row6 row4 row5
begin
row0 = u0
row10 = u10
X17 = u1 + u13
X18 = u14 + u16
row8 = u8 + X18
X20 = u2 + u4
row2 = u3 + X20
X22 = u6 + X17
X23 = u5 + X22
X24 = u11 + X23
X25 = u12 + X23
X26 = u7 + X25
X27 = u8 + X25
row7 = row2 + X27
X29 = row8 + X24
row3 = u3 + X29
X31 = u6 + u10
row9 = u14 + X31
X33 = u1 + u2
row1 = u0 + X33
X35 = u4 + X26
X36 = X33 + X35
X37 = u14 + u15
row6 = X36 + X37
X39 = row7 + X35
X40 = row3 + X39
X41 = u6 + u9
row4 = X40 + X41
X43 = X26 + X31
X44 = X17 + X24
X45 = u0 + X44
row5 = X43 + X45
end
```

Extended matrix The following is a straight-line program computing the extended matrix from the 8-way split in Section 4.4.

```
59 gates
34 inputs
u0 u1 u2 u3 u4 u5 u6 u7 u8 u9 u10 u11 u12 u13 u14 u15 u16
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16
```

```
10 outputs
row1 row8 row9 row0 row7 row2 row4 row6 row3 row5
begin
TR1 = u12 + v7
TR2 = u16 + v3
TR3 = u7 + v11
TR4 = u8 + v4
X26 = v1 + v13
X27 = u13 + u6
X28 = u10 + v14
X29 = u2 + v0
X30 = u3 + v2
X31 = u14 + v8
X32 = v6 + X26
X33 = v12 + X32
X34 = u5 + X27
X35 = u1 + u11
X36 = TR2 + TR4
X37 = u4 + X29
X38 = u14 + X34
X39 = X30 + X37
row1 = v1 + X39
X41 = v5 + X33
X42 = v16 + X28
X43 = u6 + X42
row8 = X31 + X43
X45 = TR1 + TR3
X46 = X34 + X41
X47 = v6 + X28
row9 = v10 + X47
X49 = u1 + X29
row0 = u0 + X49
X51 = u2 + X46
X52 = X36 + X41
X53 = X31 + X52
row7 = v2 + X53
X55 = X36 + X38
X56 = X30 + X35
row2 = X55 + X56
X58 = X37 + X38
X59 = X33 + X58
X60 = X45 + X59
X61 = v10 + X60
X62 = X49 + X51
X63 = row1 + X62
```

```
X64 = TR4 + X63
X65 = TR1 + X64
X66 = v14 + X65
X67 = v13 + X60
X68 = row8 + row2
X69 = X67 + X68
X70 = X66 + X69
X71 = X49 + X70
X72 = X47 + X71
X73 = row7 + X69
X74 = u13 + X73
X75 = u0 + row1
X76 = X74 + X75
row4 = X76 + v9
row6 = X66 + v15
row3 = X72 + u9
row5 = X61 + u15
end
```

A.3 7-way Split

The matrices used in the 7-way split are as follows:

$$T = \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 1&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&1&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 1&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&1&0&1&0 \\ 0&1&1&0&1&1&0 \\ 1&1&0&1&1&1&0 \\ 0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&1 \\ 1&1&0&1&1&0&1 \\ 0&0&0&0&0&1&1 \\ 1&1&0&0&0&1&1 \\ 1&0&1&1&0&1&1 \\ 0&1&1&1&0&1&1 \\ 1&1&1&1&1&1&1 \end{pmatrix},$$

and

$$R = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0\\
1&1&0&1&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&0&1\\
1&0&0&1&1&0&1&1&0&0&1&0&1&1&0&1&0&0&0&0&1&1\\
0&0&1&1&0&1&1&0&1&1&0&1&0&1&0&1&0&0&0&1&0&1\\
0&0&0&1&0&1&0&1&0&1&1&0&0&1&1&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&1&0&0&1&0&1&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0
\end{pmatrix}.$$

Top matrix The following is a straight-line program computing the top matrix from the 7-way split in Section 4.4.

```
16 gates
7 inputs
a0 a1 a2 a3 a4 a5 a6
22 outputs
baseA0 baseA1 baseA2 baseA3 baseA4 baseA5 baseA6 baseA7 baseA8 baseA9
baseA10 baseA11 baseA12 baseA13 baseA14 baseA15 baseA16 baseA17 baseA18 baseA19
baseA20 baseA21
begin
baseA0 = a0
baseA1 = a1
baseA3 = a2
baseA5 = a3
baseA7 = a4
baseA9 = a5
baseA13 = a6
X7 = a0 + a1
baseA2 = X7
X8 = a0 + a2
baseA4 = X8
X9 = a0 + a4
baseA8 = X9
X10 = a1 + a3
baseA6 = X10
X11 = a2 + a6
baseA14 = X11
X12 = a3 + a5
baseA10 = X12
X13 = a4 + a6
```

```
baseA15 = X13
X14 = a5 + a6
baseA17 = X14
X15 = X7 + X14
baseA18 = X15
X16 = a1 + X12
X17 = X9 + X16
baseA12 = X17
X18 = X11 + X16
baseA20 = X18
X19 = X7 + X18
baseA19 = X19
X20 = X9 + X18
baseA21 = X20
X21 = X14 + X17
baseA16 = X21
X22 = X19 + X21
baseA11 = X22
end
```

Main matrix The following is a straight-line program computing the main matrix from the 7-way split in Section 4.4.

```
41 gates
22 inputs
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19 T20 T21
13 outputs
row0 row12 row11 row1 row10 row2 row8 row4 row3 row7 row5 row9 row6
begin
row0 = T0
row12 = T13
X22 = T9 + row12
row11 = T17 + X22
X24 = row0 + T1
row1 = T2 + X24
X26 = T5 + T7
X27 = T3 + T6
X28 = T10 + X26
X29 = T5 + X27
X30 = T15 + X22
row10 = T7 + X30
X32 = T8 + X29
X33 = T4 + X24
row2 = T3 + X33
X35 = T14 + X28
X36 = T11 + T21
```

```
X37 = T16 + T17
X38 = T2 + T19
X39 = T4 + T20
X40 = X29 + X39
X41 = T15 + X28
X42 = T12 + X41
X43 = X22 + X35
row8 = T3 + X43
X45 = X24 + X32
row4 = T7 + X45
X47 = X38 + X40
X48 = T18 + X47
row3 = T17 + X48
X50 = X36 + X38
X51 = X30 + X50
row7 = X32 + X51
X53 = X36 + X37
X54 = X33 + X53
row5 = X35 + X54
X56 = X37 + X42
X57 = T2 + T18
row9 = X56 + X57
X59 = T21 + X40
X60 = X42 + X59
X61 = row0 + X60
row6 = row12 + X61
end
```

Extended matrix The following is a straight-line program computing the extended matrix from the 7-way split in Section 4.4.

```
75 gates
44 inputs
u0 u1 u2 u3 u4 u5 u6 u7 u8 u9 u10 u11 u12 u13 u14 u15 u16 u17 u18 u19 u20 u21
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17
v18 v19 v20 v21
12 outputs
row10 row1 row0 row3 row9 row11 row7 row6 row5 row8 row4 row2
begin
PA0 = u7 + v5
PA1 = u5 + v3
PA2 = v0 + v1
PA3 = v6 + PA1
PA4 = v9 + v13
PA5 = v7 + v10
PA6 = u13 + v15
```

```
PA7 = u15 + PA5
PA8 = u9 + PA6
PA9 = u6 + v4
PA10 = u10 + PA0
PA11 = u4 + PA2
PA12 = u3 + PA9
PA13 = u0 + u1
PA14 = PA7 + PA8
PA15 = PA11 + PA12
PA16 = PA7 + PA10
PA17 = PA11 + PA13
PA18 = PA4 + PA8
PA19 = PA3 + PA12
PA20 = v18 + PA0
PA21 = PA3 + PA10
PA22 = v14 + PA16
PA23 = v17 + PA20
PA24 = v20 + PA19
PA25 = v11 + v21
PA26 = v12 + PA14
PA27 = v2 + PA23
PA28 = u17 + PA1
PA29 = u20 + PA15
PA30 = u17 + v7
PA31 = v8 + PA21
PA32 = u18 + PA28
PA33 = u14 + PA31
PA34 = u12 + PA22
PA35 = u13 + v17
PA36 = u11 + u21
PA37 = u8 + PA24
PA38 = u2 + PA32
PA39 = u3 + v2
PA40 = u2 + v0
PA41 = PA37 + PA40
PA42 = PA36 + PA41
PA43 = PA34 + PA38
PA44 = PA33 + PA36
PA45 = PA33 + PA39
PA46 = PA34 + PA35
PA47 = PA29 + PA38
PA48 = PA29 + PA46
PA49 = PA25 + PA48
PA50 = PA30 + PA44
PA51 = PA27 + PA37
```

```
PA52 = PA25 + PA45
PA53 = PA26 + PA42
PA54 = PA26 + PA27
PA55 = PA18 + PA52
row10 = PA18 + PA30
PA57 = PA17 + PA50
row1 = PA17 + PA39
PA59 = v21 + PA53
row0 = PA13 + PA40
PA61 = PA13 + PA51
PA62 = PA4 + PA43
row3 = v19 + PA61
row9 = v16 + PA54
row11 = PA4 + PA35
PA66 = v16 + PA49
row7 = v19 + PA55
PA68 = v13 + PA59
row6 = u19 + PA68
PA70 = u21 + PA66
row5 = u0 + PA70
row8 = u16 + PA62
row4 = u16 + PA57
row2 = u19 + PA47
end
```

B Sample Circuits for 10 and 15

B.1 $n = 10$

B.2 $n = 15$

```
312 gates
30 inputs
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14
29 outputs
C0 C1 C2 C27 C28 C26 C5 C23 C8 C20 C11 C14 C17 C24
C25 C3 C4 C21 C22 C6 C7 C15 C16 C12 C13 C18 C19 C9 C10
begin
T1 = A0 + A3
T2 = A1 + A4
T3 = A2 + A5
T4 = A0 + A6
T5 = A1 + A7
T6 = A2 + A8
```

T7 = A6 + A12
T8 = A7 + A13
T9 = A8 + A14
T10 = A3 + T7
T11 = A4 + T8
T12 = A5 + T9
T13 = A9 + A12
T14 = A10 + A13
T15 = A11 + A14
T16 = A9 + T4
T17 = A10 + T5
T18 = A11 + T6
T19 = T1 + T13
T20 = T2 + T14
T21 = T3 + T15
T22 = A6 + T19
T23 = A7 + T20
T24 = A8 + T21
T25 = B0 + B3
T26 = B1 + B4
T27 = B2 + B5
T28 = B0 + B6
T29 = B1 + B7
T30 = B2 + B8
T31 = B6 + B12
T32 = B7 + B13
T33 = B8 + B14
T34 = B3 + T31
T35 = B4 + T32
T36 = B5 + T33
T37 = B9 + B12
T38 = B10 + B13
T39 = B11 + B14
T40 = B9 + T28
T41 = B10 + T29
T42 = B11 + T30
T43 = T25 + T37
T44 = T26 + T38
T45 = T27 + T39
T46 = B6 + T43
T47 = B7 + T44
T48 = B8 + T45
T49 = A0 x B0
T50 = A0 x B1
T51 = A0 x B2

T52 = B0 x A1
T53 = B0 x A2
T54 = T50 + T52
T55 = T51 + T53
T56 = A1 x B1
T57 = A1 x B2
T58 = B1 x A2
T59 = T57 + T58
T60 = A2 x B2
T61 = T55 + T56
T62 = A3 x B3
T63 = A3 x B4
T64 = A3 x B5
T65 = B3 x A4
T66 = B3 x A5
T67 = T63 + T65
T68 = T64 + T66
T69 = A4 x B4
T70 = A4 x B5
T71 = B4 x A5
T72 = T70 + T71
T73 = A5 x B5
T74 = T68 + T69
T75 = T1 x T25
T76 = T1 x T26
T77 = T1 x T27
T78 = T25 x T2
T79 = T25 x T3
T80 = T76 + T78
T81 = T77 + T79
T82 = T2 x T26
T83 = T2 x T27
T84 = T26 x T3
T85 = T83 + T84
T86 = T3 x T27
T87 = T81 + T82
T88 = A6 x B6
T89 = A6 x B7
T90 = A6 x B8
T91 = B6 x A7
T92 = B6 x A8
T93 = T89 + T91
T94 = T90 + T92
T95 = A7 x B7
T96 = A7 x B8

T97 = B7 x A8
T98 = T96 + T97
T99 = A8 x B8
T100 = T94 + T95
T101 = T4 x T28
T102 = T4 x T29
T103 = T4 x T30
T104 = T28 x T5
T105 = T28 x T6
T106 = T102 + T104
T107 = T103 + T105
T108 = T5 x T29
T109 = T5 x T30
T110 = T29 x T6
T111 = T109 + T110
T112 = T6 x T30
T113 = T107 + T108
T114 = A9 x B9
T115 = A9 x B10
T116 = A9 x B11
T117 = B9 x A10
T118 = B9 x A11
T119 = T115 + T117
T120 = T116 + T118
T121 = A10 x B10
T122 = A10 x B11
T123 = B10 x A11
T124 = T122 + T123
T125 = A11 x B11
T126 = T120 + T121
T127 = T16 x T40
T128 = T16 x T41
T129 = T16 x T42
T130 = T40 x T17
T131 = T40 x T18
T132 = T128 + T130
T133 = T129 + T131
T134 = T17 x T41
T135 = T17 x T42
T136 = T41 x T18
T137 = T135 + T136
T138 = T18 x T42
T139 = T133 + T134
T140 = A12 x B12
T141 = A12 x B13

T142 = A12 x B14
T143 = B12 x A13
T144 = B12 x A14
T145 = T141 + T143
T146 = T142 + T144
T147 = A13 x B13
T148 = A13 x B14
T149 = B13 x A14
T150 = T148 + T149
T151 = A14 x B14
T152 = T146 + T147
T153 = T7 x T31
T154 = T7 x T32
T155 = T7 x T33
T156 = T31 x T8
T157 = T31 x T9
T158 = T154 + T156
T159 = T155 + T157
T160 = T8 x T32
T161 = T8 x T33
T162 = T32 x T9
T163 = T161 + T162
T164 = T9 x T33
T165 = T159 + T160
T166 = T10 x T34
T167 = T10 x T35
T168 = T10 x T36
T169 = T34 x T11
T170 = T34 x T12
T171 = T167 + T169
T172 = T168 + T170
T173 = T11 x T35
T174 = T11 x T36
T175 = T35 x T12
T176 = T174 + T175
T177 = T12 x T36
T178 = T172 + T173
T179 = T13 x T37
T180 = T13 x T38
T181 = T13 x T39
T182 = T37 x T14
T183 = T37 x T15
T184 = T180 + T182
T185 = T181 + T183
T186 = T14 x T38

T187 = T14 x T39
T188 = T38 x T15
T189 = T187 + T188
T190 = T15 x T39
T191 = T185 + T186
T192 = T19 x T43
T193 = T19 x T44
T194 = T19 x T45
T195 = T43 x T20
T196 = T43 x T21
T197 = T193 + T195
T198 = T194 + T196
T199 = T20 x T44
T200 = T20 x T45
T201 = T44 x T21
T202 = T200 + T201
T203 = T21 x T45
T204 = T198 + T199
T205 = T22 x T46
T206 = T22 x T47
T207 = T22 x T48
T208 = T46 x T23
T209 = T46 x T24
T210 = T206 + T208
T211 = T207 + T209
T212 = T23 x T47
T213 = T23 x T48
T214 = T47 x T24
T215 = T213 + T214
T216 = T24 x T48
T217 = T211 + T212
T218 = T61 + T74
T219 = T87 + T218
T220 = T126 + T152
T221 = T191 + T220
T222 = T100 + T113
T223 = T218 + T222
T224 = T100 + T165
T225 = T220 + T224
T226 = T139 + T217
T227 = T61 + T204
T228 = T225 + T226
T229 = T227 + T228
T230 = T178 + T219
T231 = T221 + T226

T232 = T230 + T231
T233 = T152 + T178
T234 = T204 + T217
T235 = T223 + T233
T236 = T234 + T235
T237 = T49 + T59
T238 = T54 + T60
T239 = T140 + T150
T240 = T145 + T151
T241 = T124 + T239
T242 = T125 + T240
T243 = T189 + T241
T244 = T190 + T242
T245 = T62 + T237
T246 = T67 + T238
T247 = T75 + T245
T248 = T80 + T246
T249 = T88 + T72
T250 = T93 + T73
T251 = T114 + T98
T252 = T119 + T99
T253 = T241 + T251
T254 = T242 + T252
T255 = T179 + T253
T256 = T184 + T254
T257 = T163 + T255
T258 = T164 + T256
T259 = T245 + T249
T260 = T246 + T250
T261 = T85 + T259
T262 = T86 + T260
T263 = T101 + T261
T264 = T106 + T262
T265 = T166 + T137
T266 = T171 + T138
T267 = T205 + T215
T268 = T210 + T216
T269 = T265 + T267
T270 = T266 + T268
T271 = T249 + T251
T272 = T250 + T252
T273 = T111 + T271
T274 = T112 + T272
T275 = T153 + T273
T276 = T158 + T274

T277 = T263 + T269
T278 = T264 + T270
T279 = T192 + T243
T280 = T197 + T244
T281 = T176 + T277
T282 = T177 + T278
T283 = T279 + T281
T284 = T280 + T282
T285 = T257 + T269
T286 = T258 + T270
T287 = T247 + T285
T288 = T248 + T286
T289 = T202 + T287
T290 = T203 + T288
T291 = T127 + T289
T292 = T132 + T290
T293 = T239 + T275
T294 = T240 + T276
T295 = T215 + T293
T296 = T216 + T294
T297 = T59 + T176
T298 = T60 + T177
T299 = T295 + T297
T300 = T296 + T298
T301 = T202 + T299
T302 = T203 + T300
T303 = T237 + T295
T304 = T238 + T296
T305 = T267 + T303
T306 = T268 + T304
T307 = T150 + T305
T308 = T151 + T306
T309 = T127 + T307
T310 = T132 + T308
T311 = T192 + T309
T312 = T197 + T310
C0 = T49
C1 = T54
C2 = T61
C27 = T150
C28 = T151
C26 = T152
C5 = T219
C23 = T221
C8 = T223

C20 = T225
C11 = T229
C14 = T232
C17 = T236
C24 = T243
C25 = T244
C3 = T247
C4 = T248
C21 = T257
C22 = T258
C6 = T263
C7 = T264
C15 = T283
C16 = T284
C12 = T291
C13 = T292
C18 = T301
C19 = T302
C9 = T311
C10 = T312
end